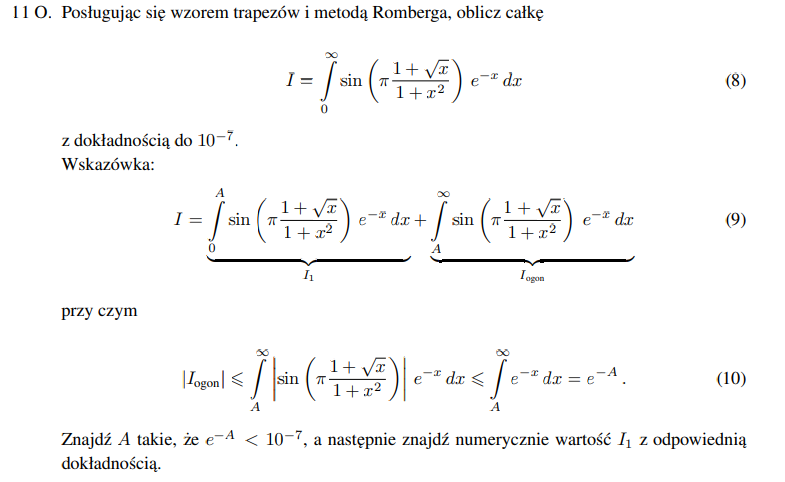
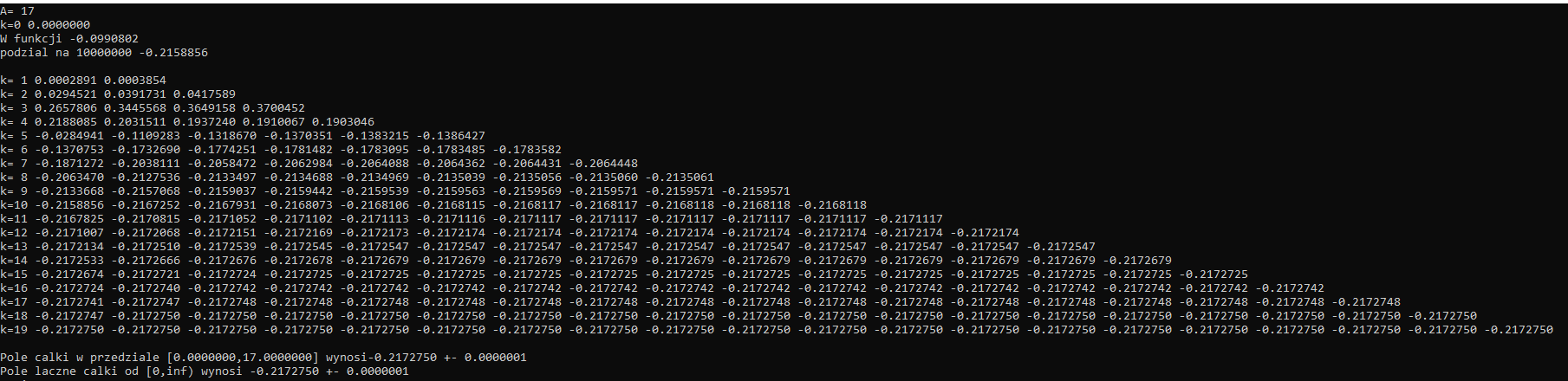
**Zadanie 11:**



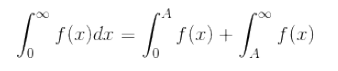
Kod w języku C++:

|  |
| --- |
| #include<iostream> #include<cmath> #include<vector> #include<iomanip>   using namespace std; long double trapezoidArea(long double,long double,int); long double calcFunction(long double);  int main(){   long double tolerance = pow(10, -7);   int i = -1;  long double rest=0;   while(true){  rest=std::pow(M\_E,i);  if(rest < tolerance)  break;  i--;  }  cout << "A= "<<-i<<endl;   //gorna granica calki  long double upper=-i;  //dolna granica calki  long double lower = 0;  //limit iteracji  long double maxIters = 32;   std::vector<long double> rombergs; //obliczanie przyblizen do obecnego wiersza  std::vector<long double> prev\_rombergs; //trzyma poprzednie przyblizenia(poprzedni wiersz)   //ustawianie precyzji  cout<<std::fixed<<std::setprecision(7);   //oblicz pierwszy wyraz  rombergs.push\_back(trapezoidArea(lower, upper, 0));  cout<<"k="<<0<<" "<<rombergs[0]<<endl;  cout<<"W funkcji "<<calcFunction(0.001)<<endl;  cout<<"podzial na 10000000 "<<trapezoidArea(0,17,10)<<endl;   int tempK;  long double temp;  int k;   for(k=1; k<maxIters-1; k++){   //wczesniej obliczony wiersz staje sie prev\_rombergs-kopiowanie  prev\_rombergs=rombergs;  //czyscimy obecny wektor  rombergs.clear();   if(k<10)  cout<<"k= "<<k<<" ";  else  cout <<"k="<<k<<" ";   tempK=k;   for(int n=0; n<=k; n++){  if(n==0){  rombergs.push\_back(trapezoidArea(lower,upper,k));  cout<<rombergs[0]<<" ";  tempK--;  }  else{  temp=(std::pow(4,n)\*rombergs[n-1]-prev\_rombergs[n-1])/(std::pow(4,n)-1);  cout<<temp<<" ";  rombergs.push\_back(temp);  tempK--;  }  }  cout<<endl;   if(std::abs(rombergs[k]-prev\_rombergs[k-1]) < tolerance){  cout<<"Pole calki w przedziale ["<<lower<<","<<upper<<"] wynosi"<<rombergs[k]<<" +- "<<tolerance<<endl;  break;  }  }   long double result=rombergs[k]+rest;  cout<<"Pole laczne calki od [0,inf) wynosi "<<result<<" +- "<<tolerance<<endl;   return 0; }  //korzysta z zlozonej metody trapezow by podzielic calke na k przedzialow long double trapezoidArea(long double *low*, long double *high*, int *k*){   //ilosc punktow podzialu  long double N=pow(2,*k*);  long double area=0;   //przedzial pomiedzy argumentami funkcji  long double h=(*high*-*low*)/N;   for(long double i=*low*; i<=*high*; i=i+h){   if(i==*low* || i==*high*)  area+=calcFunction(i)/2;  else  area+=calcFunction(i);  }  return area\*h; }  //oblicza wartosc funkcji podcalkowej dla danego x long double calcFunction(long double *x*){  return sin(M\_PI\*(1+sqrt(*x*))/(1+pow(*x*,2)))\*std::pow(M\_E,-*x*); } |

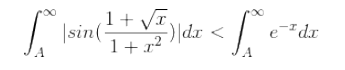
Wynik działania programu:

Opis rozwiązania:

Metoda Romberga to iteracyjna metoda obliczania coraz bliższego przybliżenia całki, która korzysta z ekstrapolacji Richardsona i złożonej metody trapezów. Nie zawsze ona działa, liczba iteracji może być nieskończona a całka rozbieżna. W naszym przykładzie całka zmierza od zera do nieskończoności więc nie możemy jej bezpośrednio obliczyć numerycznie. Całkę należy rozbić na dwa przedziały:



W zależności od dokładności możemy zauważyć, że całka nie będzie nigdy większa niż e-x .



Możemy wyszacować analitycznie i przybliżyć całkę e-x, która będzie mniejsza niż zadana dokładność. Dzięki temu będzie możliwe określenie przedziału [0, A], w którym obliczanie całki będzie miało znaczący wpływ na wynik końcowy. Dla dokładności 10-7 A wyniosło 17

Omówienie wyników

Obliczone A to granica całki, od której dalszy wynik nie wpływa znacząco na całościowe pole

Co udowadnia:

Obraz zawierający obiekt

Opis wygenerowany automatycznie

Metoda Romberga osiąga oczekiwaną znacznie szybciej niż gdybyśmy próbowali korzystać bezpośrednio z metody trapezów.